Convergent decomposition techniques for training RBF neural networks.

LETTER **Communicated by John Platt**

Convergent Decomposition Techniques for Training RBF Neural Networks

C. Buzzi
L. Grippo
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza," Via Buonarroti 12 00185, Roma, Italy

M. Sciandrone
Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30 - 00185 Roma, Italy

In this article we define globally convergent decomposition algorithms for supervised training of generalized radial basis function neural networks. First, we consider training algorithms based on the two-block decomposition of the network parameters into the vector of weights and the vector of centers. Then we define a decomposition algorithm in which the selection of the center locations is split into sequential minimizations with respect to each center, and we give a suitable criterion for choosing the centers that must be updated at each step. We prove the global convergence of the proposed algorithms and report the computational results obtained for a set of test problems.

1 Introduction

We consider a generalized radial basis function (RBF) neural network with M hidden nodes and one output unity, whose input-output mapping is defined by

$$y(x; w_1, \ldots, w_M, c_1, \ldots, c_M) = \sum_{i=1}^{M} w_i \phi(\|x - c_i\|^2),$$

where $x \in R^n$ is the input vector, $w_i \in R$, for $i = 1, \ldots, M$ are the output weights, $c_i \in R^n$, for $i = 1, \ldots, M$, are the centers, $\phi : R^+ \to R^+$ is an RBF, and $\|\cdot\|$ is the Euclidean norm.

Given a set of input-output training pairs $(x^p, t^p) \in R^n \times R$, for $p = 1, \ldots, P$, and letting $w = (w_1, \ldots, w_M)$ and $c = (c_1, \ldots, c_M)$, we define by $E_p(w, c)$ a measure of the distance between the desired output t^p corresponding to the input x^p and the actual output $y(x^p; w, c)$ of the network, so that

the overall error function E is given by

$$E(w, c) = \sum_{p=1}^{P} E_p(w, c).$$

Two basic approaches are commonly adopted for tackling the learning problem in generalized RBF networks (Bishop, 1995; Haykin, 1999): (1) unsupervised selection of centers and supervised learning of the output weights and (2) supervised learning of both center positions and output weights. The first strategy consists of fixing the center positions c_i (and possibly the other internal parameters of each unit) using an unsupervised technique and then estimating the linear weights w_i of the output layer by means of a supervised technique consisting of minimizing E with respect to w, which typically corresponds to solving a linear least-squares problem. The simplest criterion for choosing the center positions can be that of setting them equal to a random subset of the input vectors from the training set. As an improvement on this procedure several clustering techniques have been also suggested, such as the K-means clustering algorithm of Moody and Darken (1989), which attempt to estimate appropriate locations for the centers that more accurately reflect the distribution of the data points (see Bishop, 1995, and Haykin, 1999, for a review). We note that the use of unsupervised techniques takes no account of the target values associated with the training inputs, and it may lead to the use of an unnecessarily large number of basis functions in order to achieve adequate performance.

The alternative training strategy consists of the supervised selection of both center positions and output weights (Poggio & Girosi, 1990), and it can be implemented by minimizing the training error E with respect to both w and c using gradient-descent techniques. Some computational experimentation performed on the NETtalk task (Wettschereck & Dietterich, 1992) has shown that generalized RBF networks with supervised selection of center positions yield improved generalization performance, in comparison with RBF based on unsupervised selection of centers and supervised learning of the output weights. However, choosing the basis function parameters by supervised learning may constitute a difficult nonlinear optimization problem, which can be quite computationally intensive for large values of the input dimension and of the size of the training set.

In order to overcome this difficulty to some extent, while retaining the advantages of supervised learning, we can attempt to decompose the optimization problem into a sequence of simpler problems by partitioning the problem variables into different blocks and then employing some block-descent technique, based on alternate partial minimizations with respect to the different blocks. This permits the use of specialized techniques for the solution of the subproblems, which can be advantageous in terms of both computing times and error values. However, appropriate conditions should be satisfied in order to guarantee convergence toward minimizers

of the error function. In fact, block descent methods may not converge in the nonconvex case unless suitable precautions are adopted for avoiding oscillatory behavior (Bertsekas, 1999; Powell, 1973).

In this article, on the basis of some recent results on block descent methods for nonlinear optimization (Grippo & Sciandrone, 1999, 2000), we define convergent decomposition algorithms for supervised training of generalized RBF networks. First, we consider a two-block training scheme, based on alternating the optimization with respect to the output weights with the optimization of the center locations. In this scheme, as in the case of unsupervised learning, we retain the possibility of computing a global optimum with respect to the output weights by solving a least-squares problem, but we can also guarantee convergence toward stationary points of the error function with respect to both c and w. However, when the number M of centers and the dimension n of the input space are very large, even the approximate minimization with respect to c can be computationally expensive. Therefore, we define a decomposition algorithm in which the optimization of the center locations is split into sequential minimizations with respect to each center c_i, and we introduce a suitable criterion for selecting the centers that must be updated at each step. In this way, unnecessary operations can be avoided; moreover, when a single center is updated, considerable computational savings can be obtained by exploiting the structure of the objective function in the minimization process. The convergence of this technique is ensured by imposing appropriate conditions on the approximate minimizations with respect to c_i.

In section 2 we formally define the learning problem for RBF networks and introduce some basic notation. In section 3 we describe two-block learning algorithms based on alternating the optimization with respect to the output weights with the optimization of the center locations, and we prove the convergence of these schemes. In section 4 we define a globally convergent learning algorithm based on the additional decomposition of the vector c into M blocks. In section 5 we show the results obtained with the proposed algorithms in the solution of some standard test problems taken from Preschelt (1994); in particular, we report the results of early stopping experiments and comparisons with an efficient reduced memory quasi-Newton algorithm of the Numerical Algorithms Group (NAG) library, in terms of both computing times and generalization errors. Finally, some concluding remarks are given in section 6. In appendixes A and B we collect the convergence proofs of the proposed algorithms. In appendix C we describe in detail a line search procedure employed in the convergence analysis and the computational experimentation.

2 Problem Formulation and Basic Notation

Given the set of input-output training pairs $(x^p, t^p) \in R^n \times R$, for $p = 1, \ldots, P$, we define the least-squares error:

$$E_{\mathrm{LS}}(w, c) = \sum_{p=1}^{P} \left(\sum_{i=1}^{M} w_i \phi(\|x^p - c_i\|^2) - t^p \right)^2,$$

where it is assumed that the function ϕ is continuously differentiable on R^+.

If we introduce the $P \times M$ matrix $\Phi(c)$ with elements

$$(\Phi(c))_{p,i} = \phi(\|x^p - c_i\|^2),$$

and we set $t = (t^1, \ldots, t^P)^T$, then the error function E_{LS} can be put into the form

$$E_{\mathrm{LS}}(w, c) = \|\Phi(c)w - t\|^2 .$$

For the solution of the training problem, we consider a regularized error function (Bishop, 1995), which is obtained by adding to E_{LS} a regularization term expressed as the squared norm of the network parameters, that is:

$$E(w, c) = E_{\mathrm{LS}}(w, c) + \epsilon(\|w\|^2 + \|c\|^2), \tag{2.1}$$

where ϵ is a given positive number. We note that the function E can be put into the form

$$E(w, c) = \left\| \begin{pmatrix} \Phi(c) \\ \sqrt{\epsilon} I \end{pmatrix} w - \begin{pmatrix} t \\ 0 \end{pmatrix} \right\|^2 + \epsilon \|c\|^2,$$

and hence, if we introduce the $(P + M) \times M$ matrix $\tilde{\Phi}(c)$ and the $(P + M)$ vector $\tilde{t}$ defined by

$$\tilde{\Phi}(c) := \begin{pmatrix} \Phi(c) \\ \sqrt{\epsilon} I \end{pmatrix}, \quad \tilde{t} := \begin{pmatrix} t \\ 0 \end{pmatrix},$$

where I is the $M \times M$ identity matrix, we can also write

$$E(w, c) = \left\| \tilde{\Phi}(c)w - \tilde{t} \right\|^2 + \epsilon \|c\|^2.$$

Given a vector $(w^0, c^0) \in R^M \times R^{Mn}$ of network parameters, we can define the level set of E corresponding to $E(w^0, c^0)$, that is:

$$\mathcal{L}_0 = \{(w, c) \in R^M \times R^{Mn} : \ E(w, c) \leq E(w^0, c^0)\}.$$

We note that the function E has the following properties:

- E is a strictly convex quadratic function of w for fixed c.

- The level set $\mathcal{L}_0$ is compact for any given $(w^0, c^0) \in R^M \times R^{Mn}$.
- E admits a global minimum point in $R^M \times R^{Mn}$.

Then the supervised training problem associated with the given data set can be formulated as the optimization problem

$$\begin{array}{l} \text{minimize } E(w, c) \\ w \in R^M, \\ c \in R^{Mn}. \end{array} \tag{2.2}$$

We will indicate by $\nabla_w E(w, c) \in R^M$, $\nabla_{c_i} E(w, c) \in R^n$, $\nabla_c E(w, c) \in R^{Mn}$ the partial gradients of $E(w, c)$ with respect to w, c_i, c, that is,

$$\nabla_w E(w, c) = 2\tilde{\Phi}(c)^T \left(\tilde{\Phi}(c)w - t\right)$$

$$\nabla_{c_i} E(w, c) = -4w_i \left[\sum_{p=1}^{P} \left(\sum_{h=1}^{M} w_h \phi(\|x^p - c_h\|^2) - t^p\right) \phi'(\|x^p - c_i\|^2)(x^p - c_i)\right] + 2\epsilon c_i$$

$$\nabla_c E(w, c) = \begin{pmatrix} \nabla_{c_1} E(w, c) \\ \nabla_{c_2} E(w, c) \\ \vdots \\ \nabla_{c_M} E(w, c) \end{pmatrix},$$

where ϕ' is the first-order derivative of ϕ.

3 Two-Block Learning Algorithms

In this section, we consider two-block training procedures, which are defined as a sequence of iterations consisting of two steps. In the first step, starting from the current estimate (w^k, c^k), we compute an estimate w^{k+1} of the output weights, holding fixed the vector c^k. In the second step, holding fixed w^{k+1}, we compute a new vector c^{k+1} that updates the center positions. In this approach, as in the case of unsupervised learning of center locations, we retain a distinction between the optimization of the output weights, which can be performed by solving a linear least-squares problem, and the construction of the hidden units. However, the activation functions of the hidden units are now computed through a nonlinear optimization technique, and the repeated alternation of the two steps yields asymptotically a stationary point in the extended space of the parameters w and c.

Different schemes can be defined in relation to the technique used for updating the vector c. A first possibility can be that of performing an approximate minimization of the error function with respect to c, using some

convergent minimization algorithm, which terminates with a new vector c^{k+1} when some stopping criterion is satisfied. This strategy corresponds to a modified version of the block nonlinear Gauss-Seidel method (see, e.g., Bertsekas, 1999) and it is described formally in the following scheme, where the stopping rule is defined through an adjustable upper bound ξ^k on the norm of the partial gradient $\nabla_c E$:

Algorithm 1
Data. $w^0 \in R^M$, $c^0 \in R^{Mn}$ and a sequence $\xi^k > 0$ such that $\xi^k \to 0$.
Step 0. Set $k = 0$.
Step 1. Compute $w^{k+1} = \arg\min_w E(w, c^k)$, by solving the linear least-squares problem

$$\begin{array}{ll} \text{minimize} & \left\| \tilde{\Phi}(c^k) w - \tilde{t} \right\|^2 . \\ w \in R^M & \end{array}$$

Step 2. Using a minimization method, compute c^{k+1} such that

$$E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c^k) \qquad \text{and} \qquad \|\nabla_c E(w^{k+1}, c^{k+1})\| \leq \xi^k.$$

Step 3. Set $k = k + 1$ and go to step 1.

Although block-coordinate methods may not converge in the general nonconvex case, Grippo and Sciandrone (1999) have shown that convergence can be established in the case of a two-block decomposition, even in the absence of any convexity assumption on the objective function. In our case, the convergence proof can be further simplified because of the fact that E is strictly convex with respect to w. Taking this into account, we can establish the following proposition, whose proof is reported in appendix A.

Proposition 1. *Let $\{(w^k, c^k)\}$ be the sequence generated by algorithm 1. Then:*

i. *$\{(w^k, c^k)\}$ has limit points.*
ii. *The sequence $\{E(w^k, c^k)\}$ converges to a limit.*
iii. *Every limit point of $\{(w^k, c^k)\}$ is a stationary point of E.*

In practice, step 2 of algorithm can be performed by using any convergent algorithm for unconstrained optimization, such as the steepest-descent method. However, a more convenient approach could be that of performing only a fixed number of minimization steps with respect to c. In this case, in order to define a convergent algorithm, we must guarantee that appropriate conditions are satisfied at each iteration. Sufficient convergence conditions could be imposed, for instance, by choosing as a search direction in R^{Mn} the negative partial gradient with respect to c, that is,

$$d^k = -\nabla_c E(w^{k+1}, c^k),$$

and then requiring that

$$E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c^k + \alpha^k d^k),$$

where the step size α^k along d^k is computed by means of a "convergent" line search algorithm.

More specifically, we can require that the step size α^k is produced by a line search along d^k, such that the following condition holds:

Condition 1. *For all k we have:*

$$E(w^{k+1}, c^k + \alpha^k d^k) \leq E(w^{k+1}, c^k). \tag{3.1}$$

Moreover, if

$$\lim_{k\to\infty} E(w^{k+1}, c^k) - E(w^{k+1}, c^k + \alpha^k d^k) = 0, \tag{3.2}$$

then for every subsequence $\{(w^{k+1}, c^k)\}_K$ *converging to some* $(\bar{w}, \bar{c})$ *we have*

$$\lim_{k\in K, k\to\infty} \nabla_c E(w^{k+1}, c^k) = 0. \tag{3.3}$$

Condition 1 imposes, in essence, that the line search algorithm must be able to drive to zero the directional derivative of E along d^k (and hence the partial gradient $\nabla_c E(w^{k+1}, c^k)$), at least when equation 3.2 holds and the sequence $\{(w^{k+1}, c^k)\}$ has limit points. The steepest descent direction appearing in condition 1 can also be replaced with any "gradient-related" (Bertsekas, 1999) search direction.

Many different algorithms are available for determining a step size α^k in a way that condition 1 is satisfied. In fact, we can introduce simple modifications in the convergence proofs of standard line search algorithms (see, e.g., Bertsekas, 1999) for taking into account the fact that the objective function is now also dependent on w^k. The simplest idea could be that of employing an Armijo-type inexact line search algorithm, which consists of choosing

$$\begin{aligned}\alpha^k = \max_{j=0,1,\ldots} \{(\delta)^j \rho^k : E(w^{k+1}, c^k + (\delta)^j \rho^k d^k) \\ \leq E(w^{k+1}, c^k) - \gamma (\delta)^j \rho^k \|d^k\|^2\},\end{aligned} \tag{3.4}$$

where $\rho^k \geq \rho > 0$ is some initial estimate of the step size, $\delta \in (0, 1)$, and $\gamma \in (0, 1)$. More generally, we can adopt acceptability conditions for the step size that ensure that the objective function has been "sufficiently reduced" and the step size is "sufficiently large." In equation 3.4, the first requirement is satisfied through the condition

$$E(w^{k+1}, c^k + \alpha^k d^k) \leq E(w^{k+1}, c^k) - \gamma \alpha^k \|d^k\|^2,$$

while the second is implicitly imposed by requiring both that the initial tentative stepsize is bounded away from zero ($\rho^k \geq \rho > 0$) and that the contraction factor δ is fixed. Another particular example, where the preceding conditions are taken into account, can be derived, as a special case, from the line search algorithm described in appendix C. Making use of condition 1, we can define the following training scheme:

Algorithm 2
Data. $c^0 \in R^{Mn}$.
Step 0. Set $k = 0$.
Step 1. Compute $w^{k+1} = \arg\min_w E(w, c^k)$, by solving the linear least-squares problem:

$$\begin{array}{ll} \text{minimize} & \left\| \tilde{\Phi}(c^k) w - \tilde{t} \right\|^2 . \\ w \in R^M. & \end{array}$$

Step 2. If $\|\nabla_c E(w^{k+1}, c^k)\| = 0$ then stop; otherwise

a. Set $d^k = -\nabla_c E(w^{k+1}, c^k)$, compute α^k using any line search algorithm satisfying condition 1.

b. Choose c^{k+1} as any vector such that

$$E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c^k + \alpha^k d^k).$$

Step 3. Set $k = k + 1$ and go to step 1.

The following proposition, whose proof is reported in appendix A, states that algorithm 2 retains essentially the same convergence properties of algorithm 1:

Proposition 2. *Suppose that algorithm 2 generates an infinite sequence* $\{(w^k, c^k)\}$. *Then:*

i. $\{w^k, c^k)\}$ *has limit points.*

ii. *The sequence* $\{E(w^k, c^k)\}$ *converges to a limit.*

iii. *Every limit point of* $\{(w^k, c^k)\}$ *is a stationary point of* E.

The condition of step 2b can be satisfied, for instance, by taking $c^{k+1} = c^k + \alpha^k d^k$. More generally, starting from the point $c^k + \alpha^k d^k$, some iterations of any descent method could be performed to generate c^{k+1}. Thus, as conditions a and b of step 2 are usually met in most of computational implementations of unconstrained optimization methods (such as conjugate gradient methods or quasi-Newton methods), the preceding analysis ensures that a convergent two-block training scheme can be realized, in practice, by running some standard code for a fixed number of iterations each time that w^k is updated through a least-squares technique.

4 A Block Decomposition Algorithm

In this section we define a decomposition algorithm in which the problem variables (w, c) are partitioned into the $M + 1$ blocks corresponding to the weights w and the center positions c_i, for $i = 1, \ldots, M$. As in the algorithms of the preceding section, the weights are computed by solving a linear least-squares problem for fixed values of c, but now the center positions are updated in sequence for $i = 1, \ldots, M$. As the problem is nonconvex in (w, c) and the variables are partitioned into a number of blocks larger than two, suitable restrictions must be imposed on the computation of the updated values for c_i, in order to guarantee the convergence of the iterative process. In fact, to prevent the occurrence of oscillatory behavior, we must ensure that

$$\|c_i^{k+1} - c_i^k\| \to 0, \quad \text{for } i = 1, \ldots, M. \tag{4.1}$$

We can again use a line search along the steepest-descent direction; however, in this case, the line search algorithm must also be compatible with equation 4.1. More specifically, for each $k \geq 0$ and $i = 1, \ldots, M$, letting

$$d_i^k = -\nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k),$$

we suppose that a line search along d_i^k computes a step size α_i^k, such that the following condition holds:

Condition 2. *For all $k \geq 0$ and for all $i = 1, \ldots, M$ it results:*

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) \leq E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k). \tag{4.2}$$

Moreover, if

$$\lim_{k\to\infty} \; E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) - E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) = 0, \tag{4.3}$$

then:

i. *We have*

$$\lim_{k\to\infty} \alpha_i^k \|d_i^k\| = 0. \tag{4.4}$$

ii. *For every subsequence $\{(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)\}_K$ converging to some $(\bar{w}, \bar{c}_1, \ldots, \bar{c}_i, \ldots, \bar{c}_M)$ we have*

$$\lim_{k\in K, k\to\infty} \nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) = 0. \tag{4.5}$$

In comparison with condition 1, we note that condition 2 imposes similar requirements on the objective function reduction and on the limit behavior of the partial gradients, but also requires that equation 4.4 is satisfied.

We note that this condition could be satisfied in principle by imposing a constant upper bound on the step size α_i^k in the Armijo-type line search algorithm (algorithm 2). A more flexible line search algorithm that satisfies condition 2 without imposing a priori upper or lower bounds on the step size is described in detail in appendix C.

A convergent training algorithm, where condition 2 is taken into account, is defined in the following scheme:

Algorithm 3
Data. $c_1^0, \ldots, c_M^0 \in R^n$, $\tau_i > 0$, and sequences $\xi_i^k > 0$ such that $\xi_i^k \to 0$ for $i = 1, \ldots, M$.
Step 0. Set $k = 0$.
Step 1. Compute $w^{k+1} = \arg\min_w E(w, c^k)$, by solving the linear least-squares problem:

$$\begin{array}{ll} \text{minimize} & \left\| \tilde{\Phi}(c^k) w - \tilde{t} \right\|^2 . \\ w \in R^M. & \end{array}$$

Step 2. For $i = 1, \ldots, M$; if $\|\nabla_{c_i} E(w^{k+1}, c^k)\| \le \xi_i^k$, then set $c_i^{k+1} = c_i^k$; otherwise:

a. Set $d_i^k = -\nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots c_M^k)$; compute α_i^k using any line search algorithm satisfying condition 2.

b. Compute $\tilde{c}_i^k$ such that:

$$E(w^{k+1}, c_1^{k+1}, \ldots, \tilde{c}_i^k, \ldots, c_M^k) \le E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k);$$

if $E(w^{k+1}, c_1^{k+1}, \ldots, \tilde{c}_i^k, \ldots, c_M^k) \le E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) - \tau_i \|\tilde{c}_i^k - c_i^k\|^2$, then set $c_i^{k+1} = \tilde{c}_i^k$ else set $c_i^{k+1} = c_i^k + \alpha_i^k d_i^k$.

Step 3. Set $k = k + 1$ and go to step 1.

Remark 1. When the condition $\|\nabla_{c_i} E(w^{k+1}, c^k)\| \le \xi_i^k$ at step 2 is satisfied, the center c_i is not updated. In fact, if the gradient is "small," the minimization with respect to c_i is not expected to produce a significant reduction of the error function, and hence we can speed up the training process by avoiding unnecessary operations. A heuristic rule for the choice of the sequence $\{\xi_i^k\}$ is given in section 5, where implementation details of algorithm 3 are described. It is possible, in principle, to adopt alternative rules for selecting at each iteration the centers to be updated. In order to ensure the

convergence of the sequence, it can be sufficient, for instance, to guarantee that each center will be considered within a fixed number of successive iterations (Grippo & Sciandrone, 1999).

The conditions of step 2b, on the one hand, guarantee a sufficient reduction of the objective function and, on the other, ensure that the distance $\|c_i^{k+1} - c_i^k\|$ between successive points goes to zero. These conditions can be satisfied, for instance, by setting

$$c_i^{k+1} = c_i^k + \alpha_i^k d_i^k.$$

However, any minimization method can be adopted to generate the candidate point $\tilde{c}_i^k$.

Remark 2. We note that at step 2, when a center c_i must be updated, at least one evaluation of the error function in correspondence to new positions of c_i must be performed. However, the evaluation of the error function, which depends on the outputs $y(x^p; w^{k+1}, c_1^{k+1}, \ldots, c_i, \ldots, c_M^k)$ corresponding to input patterns x^p for $p = 1, \ldots, P$, can be organized in a way that considerable computational savings are obtained at the expense of storing the outputs obtained in correspondence to c_i^k. In fact, we can write

$$\begin{aligned} &y(x^p; w^{k+1}, c_1^{k+1}, \ldots, c_i, \ldots, c_M^k) \\ &\quad = y(x^p; w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) - w_i^k \phi(\|x^p - c_i^k\|^2) + w_i^k \phi(\|x^p - c_i\|^2), \end{aligned}$$

so that the new output is obtained by means of few elementary operations.

The convergence properties of algorithm 3 are given in the following proposition whose proof is reported in appendix B.

Proposition 3. *Let* $\{(w^k, c^k)\}$ *be the sequence generated by algorithm 3. Then:*

i. $\{(w^k, c^k)\}$ *has limit points.*

ii. *The sequence* $\{E(w^k, c^k)\}$ *admits a limit.*

iii. *We* have $\lim_{k\to\infty} \|w^{k+1} - w^k\| = 0$.

iv. *For* $i = 1, \ldots, M$, *we have* $\lim_{k\to\infty} \|c_i^{k+1} - c_i^k\| = 0$.

v. *Every limit point of* $\{(w^k, c^k)\}$ *is a stationary point of* E.

5 Computational Experiments

In this section we present the numerical results obtained with algorithms 2 and 3 in a set of four test problems taken from Preschelt (1994), and we compare the performance of the proposed algorithms with that of an efficient algorithm (routine E04DGF of NAG library), which implements a

reduced-memory quasi-Newton method for the joint minimization of the error function with respect to w and c.

We adopted as basis function ϕ the direct multiquadric function defined as

$$\phi(\|x-c\|^2) = (\|x-c\|^2 + \sigma)^{1/2},$$

where the scaling parameter σ has been chosen equal to 0.01. The regularization parameter ϵ in equation 2.1 has been fixed to the value 10^{-6}.

In what follows we report a short description of the problems, the implementation description of the training algorithms, and the computational results.

5.1 Training Problems.

5.1.1 Problem P1 (Credit Card Problem). In this classification problem, the task is that of predicting the approval or nonapproval of a credit card to a customer. The data set consists of 690 pairs (x^p, d^p), where $x^p \in R^{51}$ and $d^p \in \{0, 1\}$.

5.1.2 Problem P2 (Heart Problem). In this classification problem, the task is that of deciding whether at least one of four major vessels of the heart is reduced in diameter by more than 50%. The diagnosis must be made on the basis of personal data (e.g., age and smoking habits). The data set consists of 303 pairs (x^p, d^p), where $x^p \in R^{35}$ and $t^p \in \{0, 1\}$.

5.1.3 Problem P3 (Cancer Problem). In this classification problem, the task is that of predicting a breast lump as either benign or malignant. This classification must been performed on the basis of information regarding, for instance, the uniformity of cell size and cell shape. The data set consists of 699 pairs (x^p, t^p), where $x^p \in R^9$ and $t^p \in \{0, 1\}$.

5.1.4 Problem P4 (Building Problem). In this approximation problem, the task is that of predicting energy consumption in a building taking into account the outside temperature, outside air humidity, and other similar information. The data set consists of 4208 pairs (x^p, t^p), where $x^p \in R^{14}$ and $t^p \in R$.

5.2 Training Algorithms.

5.2.1 Algorithm Q-N. The algorithm makes use of the E04DGF routine of the NAG library. This routine implements a preconditioned, limited memory (two-step) quasi-Newton conjugate gradient method. It is intended for use on large-scale problems; in fact, it does not require matrix operations and has storage costs that grow linearly with the number of variables. It can be considered quite efficient in terms of computational cost, because of the

"good" convergence properties of quasi-Newton methods. A description of the method can be found in Gill and Murray (1979) and Gill, Murray, and Wright (1981). We have used the default values of the parameters in the E04DGF routine.

5.2.2 Algorithm 2 (Section 3). The updating of the output weight vector w (step 1) is performed by using a standard routine (F04JAF routine of NAG library) for solving the linear least-squares problem. This routine, which implements a direct method, computes the minimum norm solution of a linear least-squares problem by means of the pseudoinverse matrix (determined by the singular value decomposition). The vector of centers c is updated (step 2) by executing a fixed number of iterations (10 in our case) of the quasi-Newton method implemented by the E04DGF routine. In this case, the conditions of step 2 are satisfied automatically, as the first iteration of the quasi-Newton method consists of a line search along the negative gradient, and the objective function is reduced in the subsequent iterations. The iteration counter k is updated at step 3, so that a main iteration of algorithm 2 involves the solution of a linear least-squares problem (for the updating of the vector w) and 10 inner iterations of the E04DGF routine (for the updating of the vector c).

5.2.3 Algorithm 3 (Section 4). The vector w is updated (step 1) as in algorithm 2, while each center c_i is updated by using the line search algorithm (algorithm LS) defined in appendix C, and then letting

$$c_i^{k+1} = c_i^k + \alpha_i^k d_i^k.$$

The implementation of algorithm LS is described below.

Regarding the sequences ξ_i^k, $i = 1, \ldots, M$, we used the rule

$$\xi_i^k = \epsilon^k (1 + E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)),$$

where, starting from $\epsilon^0 = 0.5$, we set

$$\epsilon^{k+1} = \begin{cases} \epsilon^k & \text{if } c_j^{k+1} \neq c_j^k \text{ for at least one } j \in \{1, \ldots, M\} \\ 0.5\epsilon^k & \text{otherwise.} \end{cases}$$

The iteration counter k is updated at step 3, so that a single main iteration of algorithm 3 involves the solution of a linear least-squares problem (for the updating of w) and one run of algorithm LS for each center c_i that is updated.

5.2.4 Algorithm LS (Appendix C). The line search algorithm has been implemented according to the pseudocode given in appendix C with the

following values of the parameters:

$$\gamma_i^l = 10^{-4}, \quad \gamma_i^u = 10^3, \quad \theta_i^l = 2, \quad \theta_i^u = 5, \quad \delta_i^l = 10^{-2}, \quad \delta_i^u = 0.9, \quad \bar{\rho}_i = 1.$$

Regarding the initial value of the step size ρ_i^k, we set

$$\rho_i^k = \begin{cases} E_i^0 / \|d_i^0\|^2 & \text{for } k = 0 \\ \\ \alpha_i^{k-1} & \text{for } k \geq 1. \end{cases}$$

The expansion and contraction factors θ_i and δ_i are determined, when needed, on the basis of a quadratic interpolation formula, by computing first the number

$$\sigma_i = \frac{\|d_i^k\|^2 \alpha_i}{2(\|d_i^k\|^2 \alpha_i + E(w^{k+1}, \ldots, c_i^k + \alpha_i d_i^k, \ldots) - E_i^k)}$$

and then letting

$$\theta_i = \min\left[\theta_i^u, \max(\sigma_i, \theta_i^l)\right]$$

$$\delta_i = \min\left[\delta_i^u, \max(\sigma_i, \delta_i^l)\right].$$

A fine tuning of the parameters has not been performed; however, on the basis of a few experiments, it would appear that the algorithm is quite robust with respect to reasonable changes of the parameter values. We note that the value of the parameter γ_i^u may have some effect on the line search accuracy, because of the fact that for large values of γ_i^u, the first tentative step size ρ_i^k is more easily accepted, and no quadratic interpolation is performed. In some cases, in particular when a higher precision in the approximation of a stationary point is required, better results could be obtained by reducing the value of γ_i^u.

5.3 Computational Results. A first set of preliminary computational experiments was performed with the aim of evaluating the efficiency of the proposed algorithms in the minimization of function 2.1. Some computational results for all problems are reported in Buzzi, Grippo, and Sciandrone (1999). Here, we show only the comparative performance of algorithm Q-N, algorithm 2, and algorithm 3, in correspondence to problem 4 for a network with a number of hidden nodes fixed to 15 and for an increasing number P of training pairs.

In particular, we report in Table 1 the computing times (on an IBM RISC System/6000 375) required for reaching prescribed values of the normalized

Table 1: Numerical Results for Problem P4 and $M = 15$, Starting from Three Initial Points.

Training Error $E/2P$	Algorithm Q-N CPU Time (sec)			Algorithm 2 CPU Time (sec)			Algorithm 3 CPU Time (sec)		
$P = 1000$									
0.0020	25	14	18	2	2	4	1	1	1
0.0015	31	19	24	7	9	10	3	2	4
0.0010	52	29	37	18	24	20	9	9	8
$P = 2000$									
0.0020	59	40	51	4	4	4	1	2	2
0.0015	81	44	65	15	20	19	6	4	10
0.0010	126	90	98	54	71	67	21	47	43
$P = 4208$									
0.0040	108	65	111	8	8	8	4	4	4
0.0030	130	80	124	15	9	8	6	4	4
0.0020	219	186	292	115	100	64	40	40	50

error function $E/2P$ for various values of P and three different starting points (w^0, c^0), where the components of w^0 are randomly chosen in the interval $[-0.5, 0.5]$, and the vectors c_i are randomly chosen among the input training vectors. In the table, each column corresponds to the same initial point.

Note that algorithms 2 and 3 are consistently faster (in terms of CPU time) than algorithm Q-N for all values of P in almost all cases, in spite of the fact that the convergence rate of decomposition algorithms is typically much inferior to that of standard methods. Also, the most relevant savings are obtained with algorithm 3, in particular when we do not require a great precision in the objective function value. Similar behaviour of the algorithms was observed with reference to the other problems (Buzzi et al., 1999).

Starting from these results, we have performed a more extensive experimentation in order to evaluate both the computational efficiency of the proposed algorithms and the generalization properties of the trained networks. To this aim, we have adopted an early stopping strategy. Then, for each problem, we have subdivided the available data into three disjoint sets: training, validation, and test sets. As suggested in Preschelt (1994), we used in all experiments the first 50% of the data as training set, the next 25% for the validation set, and the final 25% for the test set. No preprocessing of the training data has been performed.

By using training and validation sets, we have employed the following rules to establish when the training process can be terminated:

Early Stopping Rules

- The error on the validation set has been evaluated:

Table 2: Numerical Results for Problem P1, Starting from Three Initial Points.

Algorithm	Training Set			Validation Set			Test Set			Classification Error (%)			CPU Time (sec)		
							$M = 4$								
Q-N	0.21	0.21	0.21	0.17	0.17	0.17	0.21	0.20	0.21	14.5	14	14	51	54	41
ALG2	0.20	0.21	0.21	0.17	0.17	0.17	0.21	0.21	0.20	14	14	13	35	33	35
ALG3	0.20	0.22	0.21	0.16	0.17	0.17	0.21	0.22	0.21	14	14	14	13	12	12
							$M = 10$								
Q-N	0.19	0.19	0.19	0.16	0.16	0.16	0.2	0.2	0.2	13.4	14	13.4	142	128	113
ALG2	0.19	0.19	0.18	0.17	0.17	0.17	0.2	0.21	0.19	12.2	14.5	12.	95	81	122
ALG3	0.2	0.2	0.2	0.16	0.16	0.16	0.21	0.2	0.2	14	12.7	13.	34	40	34
							$M = 20$								
Q-N	0.17	0.18	0.18	0.17	0.17	0.17	0.2	0.19	0.2	12.2	12.2	12.	350	465	277
ALG2	0.19	0.19	0.19	0.19	0.18	0.17	0.21	0.21	0.2	13.4	14	14.	174	203	169
ALG3	0.19	0.2	0.2	0.18	0.17	0.16	0.22	0.21	0.2	14	13.3	12.	59	62	61

  - — Every main iteration of algorithm 2.
  - — Every 10 main iterations of algorithm 3.
  - — Every 10 iterations of algorithm Q-N.
- Each algorithm has been stopped when the error on the validation set has failed to decrease, with respect to the best value attained, after five evaluations of the validation error.

Then the generalization capability of the network has been evaluated using the test set.

For each problem, three different architectures (in the number M of hidden nodes) have been considered. For each problem and each architecture, we have used three different starting points (w^0, c^0), where the components of w^0 are randomly chosen in the interval $[-0.5, 0.5]$, and the vectors c_i^0 are randomly chosen among the input training vectors. In each experiment, all different algorithms have been started from the same initial point. Note, however, that the choice of w^0 does not affect the behavior of algorithms 2 and 3, which start with a global minimization of E with respect to w.

The results obtained are shown in the Tables 2 through 5, where for each problem and each architecture, we report, in correspondence to each training algorithm starting from three different initial points (each column is associated to the same initial point) the normalized final error $E/2P$ relative to the training, validation, and test set; the classification error on the test set (problems P1–P3); and the CPU time employed on an IBM RISC System/6000 375. We note that the computing times reported in the tables also include the time spent for evaluating the validation error.

It can be observed that all algorithms show comparable performances in terms of generalization errors on the test set. However, algorithms 2 and 3

Table 3: Numerical Results for Problem P2, Starting from Three Initial Points.

Algorithm	Training Set			Validation Set			Test Set			Classification Error (%)			CPU Time (sec)		
							$M = 2$								
Q-N	0.25	0.26	0.24	0.31	0.3	0.32	0.15	0.13	0.11	7.9	4	4	5	7	7
ALG2	0.23	0.23	0.24	0.32	0.32	0.32	0.11	0.12	0.12	4	3.6	4	4	5	4
ALG3	0.23	0.24	0.23	0.32	0.32	0.32	0.12	0.12	0.11	2.6	2.6	2.6	2	2	2
							$M = 5$								
Q-N	0.23	0.23	0.24	0.31	0.32	0.30	0.11	0.11	0.13	4	2.6	6.6	16	13	15
ALG2	0.22	0.22	0.23	0.33	0.33	0.32	0.12	0.12	0.12	4	1.4	2.6	11	13	11
ALG3	0.19	0.23	0.24	0.32	0.35	0.32	0.14	0.12	0.11	5.2	2.6	4	13	5	5
							$M = 8$								
Q-N	0.22	0.24	0.22	0.32	0.32	0.32	0.13	0.12	0.12	2.6	4	4	26	20	19
ALG2	0.21	0.21	0.22	0.34	0.33	0.32	0.13	0.13	0.13	1.3	2.6	4	18	21	18
ALG3	0.22	0.21	0.24	0.35	0.33	0.31	0.14	0.13	0.13	2.6	5.3	4	8	7	7

Table 4: Numerical Results for Problem P3, Starting from Three Initial Points.

Algorithm	Training Set			Validation Set			Test Set			Error (%)			CPU Time (sec)		
							$M = 4$								
Q-N	0.06	0.06	0.07	0.04	0.04	0.05	0.03	0.03	0.06	1.7	1.7	2.3	147	79	16
ALG2	0.06	0.06	0.06	0.04	0.04	0.04	0.03	0.04	0.03	1.7	1.7	1.7	10	28	6
ALG3	0.06	0.06	0.06	0.04	0.03	0.04	0.03	0.03	0.03	1.7	1.1	1.1	23	31	16
							$M = 10$								
Q-N	0.06	0.06	0.06	0.04	0.04	0.04	0.03	0.03	0.03	1.7	1.7	1.7	71	44	64
ALG2	0.05	0.05	0.06	0.03	0.04	0.04	0.03	0.03	0.03	1.7	1.7	1.7	46	26	23
ALG3	0.05	0.05	0.06	0.03	0.04	0.03	0.03	0.03	0.03	1.7	1.1	1.1	22	20	13
							$M = 15$								
Q-N	0.06	0.06	0.06	0.04	0.04	0.04	0.03	0.04	0.04	1.7	1.7	1.7	136	132	94
ALG2	0.05	0.05	0.06	0.03	0.04	0.04	0.03	0.03	0.04	1.7	1.7	1.7	37	36	48
ALG3	0.06	0.06	0.06	0.04	0.04	0.04	0.03	0.04	0.03	1.7	1.7	1.1	31	12	61

are significantly faster than algorithm Q-N in terms of computing time. The advantages are more relevant in correspondence to algorithm 3 in almost cases. In particular, taking the mean value of the CPU time employed by the algorithms over the 36 runs, algorithm 3 is about three times faster than algorithm Q-N.

6 Conclusion

Supervised selection of center locations appears to be a useful technique for training generalized RBF networks. When the problem dimensions (input dimension, number of units, size of the training set) are not very large, stan-

Table 5: Numerical Results for Problem P4, Starting from Three Initial Points.

Algorithm	Training Set			Validation Set			Test Set			CPU Time (sec)		
					$M = 8$							
Q-N	0.002	0.001	0.001	0.004	0.005	0.004	0.001	0.001	0.001	124	202	164
ALG2	0.001	0.001	0.001	0.005	0.005	0.005	0.001	0.001	0.001	233	157	135
ALG3	0.001	0.001	0.001	0.005	0.005	0.005	0.001	0.001	0.001	42	131	162
					$M = 15$							
Q-N	0.001	0.001	0.001	0.005	0.004	0.005	0.001	0.001	0.001	284	239	273
ALG2	0.001	0.001	0.001	0.006	0.005	0.006	0.001	0.001	0.001	283	283	459
ALG3	0.001	0.002	0.001	0.006	0.005	0.006	0.001	0.001	0.001	100	91	102
					$M = 20$							
Q-N	0.001	0.001	0.001	0.005	0.006	0.006	0.001	0.001	0.001	335	370	448
ALG2	0.001	0.001	0.001	0.006	0.006	0.006	0.001	0.001	0.001	288	288	385
ALG3	0.001	0.001	0.001	0.006	0.006	0.006	0.001	0.001	0.001	158	119	115

dard descent methods, such as reduced-memory quasi-Newton methods or conjugate gradient methods, can provide efficient training algorithms, in the context of some learning strategy based on regularization or on early stopping rules. A potentially useful alternative has been suggested in this article (algorithm 2): the two-block decomposition of the optimization problem with respect to weight and center locations. In our experience, this typically yields a reduction in the number of objective function evaluations and, hence, reduced training times. It is also conceivable that the global optimization performed in the weight space may prevent the training algorithm from being trapped at irrelevant local minimizers, which could be a possible shortcoming (although not revealed by our experimentation) of local optimization techniques based on the joint minimization with respect to weights and centers locations.

When the problem dimension increases and the number of training pairs is very large, the problem of developing efficient algorithms for supervised learning of center locations appears to be an important problem (Wettschereck and Dietterich, 1992) and the adoption of a block-descent technique (such as algorithm 3), based on the additional decomposition with respect to the different centers, could be promising. In particular, the use of suitable criteria for choosing the centers to be updated at each step and the possibility of exploiting the structure of the objective function for reducing the computational cost of each function evaluation when a single center is updated (as explained in remark 2) appear to be valuable features. Block descent techniques can be even more advantageous in the context of an early stopping strategy. In fact, decomposition techniques are typically faster during the early stages of the minimization process, while they possess ultimately a convergence rate that can be much inferior to that of

standard methods. Thus, if high precision is not required in the location of a minimizer, a decomposition approach may yield even more significant computational gains, in comparison with the joint optimization with respect to c and w.

The essential motivation of this article has been that of providing a theoretical foundation to the development of new training algorithms for generalized RBF based on decomposition techniques and that of reporting some preliminary computational experience. Further research and experimentation may be needed for developing more efficient training codes, and we mention some of the most important points that may deserve some attention. The first could be that of defining alternative criteria for choosing the center to be updated at each step, in place of the (rather expensive) criterion based on gradient evaluation employed in algorithm 3. In particular, one may think of using some clustering technique in this phase, for selecting the most promising center to be optimized, or else of using information based on past iterations (Grippo & Sciandrone, 1999). Another point could be that of defining suitable criteria, possibly based on monitoring the behavior of the objective function, for deciding to what extent the optimization of the center selected at some step must be continued before passing to consider a new center; in this case, the choice of the algorithm for optimizing the center location must also be considered. Finally, it could be possible to replace the solution of the least-squares problem with an incremental update of the weights, each time that a new center (or a group of centers) has been moved, by employing, for instance, some modified factorization method or some iterative descent technique in the weight space.

A deeper investigation of the computational aspects and a more extensive experimentation with larger-dimensional problems will be the object of future work.

Appendix A

Proof of Proposition 1. By the instructions of the algorithm, we have, for all k:

$$E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c^k) \leq E(w^k, c^k). \tag{A.1}$$

Then, the sequence $\{(w^k, c^k)\}$ belongs to the compact set $\mathcal{L}_0$, and this proves assertion i.

Assertion ii follows from equation A.1, recalling that E is bounded from below.

In order to prove assertion iii, let $\{(w^k, c^k)\}_K$ be a subsequence converging to some $(\bar{w}, \bar{c})$. First, we observe that by the instructions at step 2, we have for all k,

$$\|\nabla_c E(w^k, c^k)\| \leq \xi^{k-1}, \tag{A.2}$$

and hence, taking limits for $k \to \infty$, $k \in K$, and recalling that, by assumption, ξ^k tends to zero, we have

$$\nabla_c E(\bar{w}, \bar{c}) = 0. \tag{A.3}$$

Therefore, we must show only that $\nabla_w E(\bar{w}, \bar{c}) = 0$.

We note that $E(w, c)$ is a positive definite quadratic function of w, with smallest eigenvalue $\lambda_{\min}\left(\nabla^2_{ww} E(w^{k+1}, c^k)\right) \geq \epsilon$, where $\epsilon > 0$ is the parameter appearing in equation 2.1. Then we can write

$$\begin{aligned} E(w^k, c^k) = \quad & E(w^{k+1}, c^k) + \nabla_w E(w^{k+1}, c^k)^T (w^k - w^{k+1}) \\ & + \tfrac{1}{2} (w^k - w^{k+1})^T \nabla^2_{ww} E(w^{k+1}, c^k)(w^k - w^{k+1}). \end{aligned} \tag{A.4}$$

As w^{k+1} minimizes $E(w, c^k)$ we have

$$\nabla_w E(w^{k+1}, c^k) = 0, \tag{A.5}$$

and hence, by equation A.4 we can write

$$\|w^{k+1} - w^k\|^2 \leq \frac{2}{\epsilon} \left(E(w^k, c^k) - E(w^{k+1}, c^k) \right). \tag{A.6}$$

By equation A.1, recalling assertion ii, it follows that

$$\lim_{k \to \infty} \left(E(w^k, c^k) - E(w^{k+1}, c^k) \right) = 0, \tag{A.7}$$

so that equation A.6 implies

$$\lim_{k \to \infty} w^{k+1} - w^k = 0,$$

and hence

$$\lim_{k \to \infty, k \in K} (w^{k+1}, c^k) = (\bar{w}, \bar{c}). \tag{A.8}$$

Then, by equations A.5 and A.8 and the continuity of ∇E, it follows that $\nabla_w E(\bar{w}, \bar{c}) = 0$, which concludes the proof.

Proof of Proposition 2. Suppose that algorithm 2 generates an infinite sequence $\{(w^k, c^k)\}$. By the instructions of the algorithm, we have, for all k,

$$E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c^k) \leq E(w^k, c^k). \tag{A.9}$$

By employing the same arguments used in the proof of proposition 1, we can prove assertions i and ii; moreover, we can show that if $\{(w^k, c^k)\}_K$ is a subsequence converging to some $(\bar{w}, \bar{c})$, we have

$$\|\nabla_w E(\bar{w}, \bar{c})\| = 0. \tag{A.10}$$

Therefore, to prove assertion iii, we must show only that $\nabla_c E(\bar{w}, \bar{c}) = 0$. Now, by the instructions of step 2 we have for each $k \in K$,

$$E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c^k + \alpha^k d^k) \leq E(w^k, c^k), \tag{A.11}$$

where α^k is the step size produced at step 2a along the direction $d^k = -\nabla_c E(w^{k+1}, c^k)$. Thus, recalling assertion ii, we have also

$$\lim_{k\to\infty} E(w^k, c^k) - E(w^k, c^k + \alpha^k d^k) = 0. \tag{A.12}$$

Then, using condition 1 we obtain $\nabla_c E(\bar{w}, \bar{c}) = 0$, which concludes the proof.

Appendix B

Proof of Proposition 3. By the instructions of the algorithm, we have

$$\begin{aligned} &E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c_1^{k+1}, \ldots, c_i^{k+1}, \ldots, c_M^k) \\ &\leq E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) \leq E(w^k, c^k). \end{aligned} \tag{B.1}$$

Then, assertions i through iii can be proved as in the proof of proposition 1.

Now, by step 2, for each $i \in \{1, \ldots, M\}$ we have either that

$$c_i^{k+1} = c_i^k + \alpha_i^k d_i^k,$$

or that $c_i^{k+1} = \tilde{c}_i$, which implies

$$\begin{aligned} \|c_i^{k+1} - c_i^k\|^2 \leq \frac{1}{\tau_i} \Big(&E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) \\ &- E(w^{k+1}, c_1^{k+1}, \ldots, c_i^{k+1}, \ldots, c_M^k) \Big). \end{aligned}$$

In any case, reasoning on subsequences if necessary, recalling equation 4.4 of condition 2, equation B.1 and assertion ii, we have that assertion iv holds.

Finally, in order to prove assertion v, by contradiction, let us assume that there exists a sequence $\{w^k, c^k\}_K$ such that

$$\lim_{k\to\infty, k\in K} (w^k, c^k) = (\bar{w}, \bar{c})$$

and

$$\|\nabla E(\bar{w}, \bar{c})\| \neq 0.$$

Recalling assertion iii, we have that

$$\lim_{k\to\infty, k\in K} w^{k+1} = \bar{w},$$

and therefore, as $\nabla_w E(w^{k+1}, c^k) = 0$ for all k it follows that $\nabla_w E(\bar{w}, \bar{c}) = 0$. Then, we must have

$$\|\nabla_{c_i} E(\bar{w}, \bar{c})\| \geq \sigma \tag{B.2}$$

for some $i \in \{1, \ldots, M\}$ and $\sigma > 0$.

For $k \in K$ and k sufficiently large, we have $\|\nabla_{c_i} E(w^{k+1}, c^k)\| > \xi_i^k$, so that

$$\begin{aligned} &E(w^{k+1}, c^{k+1}) \leq E(w^{k+1}, c_1^{k+1}, \ldots, c_i^{k+1}, \ldots, c_M^k) \\ &\leq E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) \\ &\leq E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) \leq E(w^k, c^k). \end{aligned} \tag{B.3}$$

On the other hand, we can write

$$\begin{aligned} &\|(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) - (w^k, c^k)\| \\ &\leq \|w^{k+1} - w^k\| + \|c_1^{k+1} - c_1^k\| + \ldots + \|c_{i-1}^{k+1} - c_{i-1}^k\|, \end{aligned}$$

from which, recalling assertions iii and iv, we get

$$\lim_{k\to\infty, k\in K} (w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) = (\bar{w}, \bar{c}).$$

Finally, from equation B.3, taking into account assertion ii we obtain

$$\lim_{k\to\infty} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) - E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) = 0.$$

Then, recalling the fact that α_i^k is produced by a line search technique for which condition 2 holds, we obtain $\|\nabla_{c_i} E(\bar{w}, \bar{c})\| = 0$, which contradicts equation B.2.

Appendix C

In this appendix we describe a line search algorithm that satisfies condition 2 of section 4, under the assumption that the search direction is defined by

$$d_i^k = -\nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k).$$

The algorithm we will consider is designed in a way that the following features (which have both theoretical and computational motivations) are taken into account:

i. A "sufficient reduction" of the objective function value is enforced.

ii. The quantity $\alpha_i^k \|d_i^k\|$ is forced to zero in the limit.

iii. A "sufficiently large" step size α_i^k is guaranteed.

iv. The initial tentative step size $\rho_i^k > 0$ is not subject to "a priori" restrictions, and it can be chosen, for instance, on the basis of the past history.

v. The acceptability condition can be checked with a single new function evaluation.

vi. Efficient interpolation criteria for reducing or increasing the tentative step sizes are admitted.

The first three points are directly concerned with satisfaction of condition 2; the last points (which must be compatible with the other conditions) are essentially motivated by the need for reducing as much as possible the computational cost of each line search run.

As it will be shown, points i and ii are satisfied by imposing an acceptability condition of the form

$$E(w^{k+1}, c_1^{k+1} \ldots, c_i^k + \alpha_i d_i^k, \ldots, c_M^k) \leq E_i^k - \gamma_i^l (\alpha_i)^2 \|d_i^k\|^2, \tag{C.1}$$

where $\gamma_i^l > 0$, and we set

$$E_1^k := E(w^{k+1}, c^k) \qquad E_i^k := E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) \text{ for } i > 1. \tag{C.2}$$

The requirement of point iii is fulfilled and α_i can be accepted without any further function evaluation when, in addition to equation C.1, we have also either that α_i is greater than some fixed lower bound $\bar{\rho}_i$ or that a condition of the form

$$E(w^{k+1}, c_1^{k+1} \ldots, c_i^k + \alpha_i d_i^k, \ldots, c_M^k) \geq E_i^k - \gamma_i^u (\alpha_i)^2 \|d_i^k\|^2 \tag{C.3}$$

is satisfied for $\gamma_i^u > \gamma_i^l$. Thus, requirements i, ii, and iii can be satisfied, in principle, starting from an appropriate tentative step size ρ_i^k and performing only one function evaluation (as suggested at points iv and v).

When equation C.1 is satisfied for $\alpha_i = \rho_i^k$ but the step size is not "sufficiently large" in the sense defined above, we must increase α_i by a (variable) factor θ_i (point vi), which can be determined using a safeguarded extrapolation formula in a way that $1 < \theta_i^l \leq \theta_i \leq \theta_i^u$, where θ_i^l and θ_i^u are fixed constant values. The extrapolation can be repeated until either the current

step size α_i satisfies equation C.1 and is sufficiently large, or the new tentative step size $\theta_i\alpha_i$ produces an increase in the function value or violates equation C.1, that is, we have either

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \theta_i\alpha_i d_i^k, \ldots, c_M^k) > E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i d_i^k, \ldots, c_M^k)$$

or

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \theta_i\alpha_i d_i^k, \ldots, c_M^k) > E_i^k - \gamma_i^l\,(\theta_i\alpha_i)^2\,\|d_i^k\|^2.$$

In fact, as we shall see, each of these conditions implies that α_i was sufficiently large.

If equation C.1 is not satisfied in correspondence to the initial tentative step size $\alpha_i = \rho_i^k$, the step size must be reduced by a (variable) factor δ_i such that $\delta_i^l \le \delta_i \le \delta_i^u$, where $0 < \delta_i^l < \delta_i^u < 1$ are predetermined constant values. The factor δ_i can be determined using a safeguarded interpolation formula. By repeating, if needed, the interpolation phase, we must finally obtain a "sufficiently large" step size that satisfies equation C.1.

This procedure is defined formally in the following scheme.

Line Search Algorithm (LS)

Data: $0 < \gamma_i^l < \gamma_i^u$, $1 < \theta_i^l < \theta_i^u$, $0 < \delta_i^l < \delta_i^u < 1$, $\bar{\rho}_i > 0$.

Choose an initial step size $\rho_i^k > 0$ and set $j = 0$.

If

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \rho_i^k d_i^k, \ldots, c_M^k) \le E_i^k - \gamma_i^l(\rho_i^k)^2\|d_i^k\|^2 \qquad \text{(C.4)}$$

then

1.1) Choose $\theta_i \in [\theta_i^l, \theta_i^u]$ and set $\alpha_i = \rho_i^k$;

1.2) while

$$\begin{cases} \alpha_i < \bar{\rho}_i \\ \textbf{and} \\ E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i d_i^k, \ldots, c_M^k) < E_i^k - \gamma_i^u(\alpha_i)^2\|d_i^k\|^2 \\ \textbf{and} \\ E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \theta_i\alpha_i d_i^k, \ldots, c_M^k) \le \\ \min\Big\{E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i d_i^k, \ldots, c_M^k), \\ \qquad E_i^k - \gamma_i^l\,(\theta_i\alpha_i)^2\,\|d_i^k\|^2\Big\} \end{cases}$$

set $j = j + 1$, $\alpha_i = \theta_i \alpha_i$ and update $\theta_i \in [\theta_i^l, \theta_i^u]$;

end while

1.3) set $\alpha_i^k = \alpha_i$, $\theta_i^k = \theta_i$ and **exit**.

else

2.1) choose $\delta_i \in [\delta_i^l, \delta_i^u]$, set $\alpha_i = \delta_i \rho_i^k$ and $j = 1$;

2.2) while

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i d_i^k, \ldots, c_M^k) > E_i^k - \gamma_i^l (\alpha_i)^2 \|d_i^k\|^2 \tag{C.5}$$

set $j = j + 1$, $\alpha_i = \delta_i \alpha_i$ and update $\delta_i \in [\delta_i^l, \delta_i^u]$;

end while

2.3) set $\alpha_i^k = \alpha_i$, $\delta_i^k = \delta_i$ and **exit**.

end if

In the next proposition we show that algorithms LS is well defined and terminates in a finite number of inner steps.

Proposition 4. *Suppose that $d_i^k \neq 0$. Then algorithm* LS *determines in a finite number of inner iterations a step size α_i^k such that*

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) \leq E_i^k - \gamma_i^l (\alpha_i^k)^2 \|d_i^k\|^2 \tag{C.6}$$

and at least one of the following conditions is satisfied:

i. $\alpha_i^k \geq \bar{\rho}_i$.

ii. $E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) \geq E_i^k - \gamma_i^u (\alpha_i^k)^2 \|d_i^k\|^2$.

iii. $E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \lambda_i^k \alpha_i^k d_i^k, \ldots, c_M^k) >$

$$\min \left\{ E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k), E_i^k - \gamma_i^l \left(\lambda_i^k \alpha_i^k\right)^2 \|d_i^k\|^2 \right\}$$

where $\min\{\theta_i^l, \frac{1}{\delta_i^u}\} \leq \lambda_i^k \leq \max\{\theta_i^u, \frac{1}{\delta_i^l}\}$.

Proof. Let $d_i^k \neq 0$. Assume first that condition C.4 is satisfied. Reasoning by contradiction, suppose that the cycle at step 1.2 does not terminate in a finite number of inner iterations. Let j be the value of the inner counter at the beginning of step 1.2, and denote by $\alpha_i(j)$ and $\theta_i(j)$ the corresponding values of α_i and θ_i, respectively. Then we can write

$$\alpha_i(j) \geq \left(\theta_i^l\right)^j \rho_i^k,$$

so that as $\theta_i^l > 1$, for sufficiently large j we get a contradiction to the condition $\alpha_i(j) < \bar{\rho}_i$. Therefore, there must exist a finite integer j^* such that the inner cycle at step 1.2 terminates and at least one of the conditions i, ii, iii is satisfied with $\alpha_i^k = \alpha_i(j^*)$ and $\lambda_i^k = \theta_i(j^*)$, so that $\theta_i^l \le \lambda_i^k \le \theta_i^u$. In order to show that equation C.6 holds, we distinguish the cases $j^* = 0$ and $j^* \ge 1$. If $j^* = 0$, then equation C.6 holds with $\alpha_i^k = \rho_i^k$ because of equation C.4. If $j^* \ge 1$, then we have necessarily that

$$\begin{aligned} &E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \theta_i(j^*-1)\alpha_i(j^*-1)d_i^k, \ldots, c_M^k) \\ &\le \min\Big\{E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i(j^*-1)d_i^k, \ldots, c_M^k), \\ &\qquad E_i^k - \gamma_i^l\left(\theta_i(j^*-1)\alpha_i(j^*-1)\right)^2 \|d_i^k\|^2\Big\} \end{aligned}$$

from which, recalling that $\alpha_i^k = \alpha_i(j^*) = \theta_i(j^*-1)\alpha_i(j^*-1)$, it follows that equation C.6 is satisfied.

Now assume that equation C.4 is not satisfied. Denoting by $\alpha_i(j)$ and $\delta_i(j)$ the values of α_i and δ_i at the beginning of step 2.2, for $j \ge 1$, we have

$$\alpha_i(j) \le \left(\delta_i^u\right)^j \rho_i^k.$$

Reasoning again by contradiction, suppose that the cycle of step 2.2 does not terminate in a finite number of inner iterations, so that equation C.5 holds for every $j \ge 1$. Then, we have

$$\frac{E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i(j)d_i^k, \ldots, c_M^k) - E_i^k}{\alpha_i(j)} > -\gamma_i^l \alpha_i(j)\|d_i^k\|^2,$$

so that, taking limits for $j \to \infty$, as $\alpha_i(j)$ goes to 0, we obtain:

$$\nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)^T d_i^k \ge 0,$$

which contradicts the assumption $\nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)^T d_i^k = -\|d_i^k\|^2 < 0$. Therefore, there must exist a finite integer j^* such that equation C.6 hold with $\alpha_i^k = \alpha_i(j^*)$. Moreover, as equation C.4 is not satisfied, the tentative step size has been reduced at least by a factor $\delta_i(j)$, and hence iii holds with $\lambda_i^k = 1/\delta_i(j^*)$, so that $\frac{1}{\delta_i^u} \le \lambda_i^k \le \frac{1}{\delta_i^l}$.

Now we prove that algorithm LS satisfies condition 2.

Proposition 5. *Let $\{(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)\}$ be a given sequence in $R^{M(n+1)}$, and let $\{d_i^k\}$ be the sequence of the steepest descent directions:*

$$d_i^k = -\nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k).$$

Let α_i^k be computed by means of algorithm LS *when $d_i^k \neq 0$ and set $\alpha_i^k = 0$ whenever $d_i^k = 0$. Then*

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) \leq E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k). \quad \text{(C.7)}$$

Moreover, if

$$\lim_{k\to\infty} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) - E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) = 0, \quad \text{(C.8)}$$

then:

i. *We have* $\lim_{k\to\infty} \alpha_i^k \|d_i^k\| = 0$.

ii. *For every subsequence* $\{(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)\}_K$ *converging to a point* $(\bar{w}, \bar{c}_1, \ldots, \bar{c}_i, \ldots, \bar{c}_M)$ *we have*

$$\nabla_{c_i} E(\bar{w}, \bar{c}_1, \ldots, \bar{c}_i, \ldots, \bar{c}_M) = 0.$$

Proof. If $d_i^k = 0$, then condition C.7 is obviously true; otherwise, it follows from equation C.6 of proposition 4. In order to prove assertion i, let us define the point

$$v^k := (w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k),$$

and let

$$E_i^k = E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k).$$

Then the acceptance rule of algorithm LS ensures that

$$E_i^k - E(v^k) \geq \gamma_i^l \left(\alpha_i^k\right)^2 \|d_i^k\|^2,$$

so that the limit

$$\lim_{k\to\infty} E_i^k - E(v^k) = 0$$

implies assertion i.

Now, let $\{(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)\}_K$ be a subsequence converging to $(\bar{w}, \bar{c}_1, \ldots, \bar{c}_i, \ldots, \bar{c}_M)$. Reasoning by contradiction, we can assume that there exists $\sigma > 0$ such that

$$\|\nabla_i E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)\| \geq \sigma, \quad \text{(C.9)}$$

for all $k \in K$ and k sufficiently large since otherwise, the continuity of $\nabla_{c_i} E$ and the convergence of the sequence would imply $\nabla_{c_i} E(\bar{w}, \bar{c}_1, \ldots, \bar{c}_i, \ldots, \bar{c}_M) = 0$.

By proposition 4, for all k we have that

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) \leq E_i^k - \gamma_i^l \left(\alpha_i^k\right)^2 \|d_i^k\|^2, \tag{C.10}$$

and at least one of the following conditions is satisfied:

$$\alpha_i^k \geq \bar{\rho}_i \tag{C.11}$$

$$E(w^{k+1}, c_1^k, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) \geq E_i^k - \gamma_i^u (\alpha_i^k)^2 \|d_i^k\|^2 \tag{C.12}$$

$$\begin{aligned} &E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \lambda_i^k \alpha d_i^k, \ldots, c_M^k) \\ &> E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) \end{aligned} \tag{C.13}$$

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \lambda_i^k \alpha_i^k d_i^k, \ldots, c_M^k) > E_i^k - \gamma_i^l \left(\lambda_i^k \alpha_i^k\right)^2 \|d_i^k\|^2 \tag{C.14}$$

where $\min\{\theta_i^l, \frac{1}{\delta_i^u}\} \leq \lambda_i^k \leq \max\{\theta_i^u, \frac{1}{\delta_i^l}\}$. By assertion i and equation C.9, recalling that

$$d_i^k = -\nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k),$$

we have that equation C.11 can not hold for k sufficiently large. Then, we can assume that one of the conditions C.12, C.13, or C.14 is satisfied. By the mean value theorem, we can find points

$$\begin{aligned} u^k &= (w^{k+1}, c_1^{k+1}, , \ldots, c_i^k + \tau_i^k \alpha_i^k d_i^k, \ldots, c_M^k) \qquad \text{with } \tau_i^k \in (0,1) \\ v^k &= (w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k + \eta_i^k (\lambda_i^k \alpha_i^k - \alpha_i^k) d_i^k, \ldots, c_M^k) \qquad \text{with } \eta_i^k \in (0,1) \\ w^k &= (w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \xi_i^k \lambda_i^k \alpha_i^k d_i^k, \ldots, c_M^k) \qquad \text{with } \xi_i^k \in (0,1) \end{aligned}$$

such that we can write

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) = E_i^k + \alpha_i^k \nabla_{c_i} E(u^k)^T d_i^k$$

$$\begin{aligned} &E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \lambda_i^k \alpha_i^k d_i^k, \ldots, c_M^k) \\ &= E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \alpha_i^k d_i^k, \ldots, c_M^k) + \left(\lambda_i^k \alpha_i^k - \alpha_i^k\right) \nabla_{c_i} E(v^k)^T d_i^k \end{aligned}$$

$$E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k + \lambda_i^k \alpha_i^k d_i^k, \ldots, c_M^k) = E_i^k + \lambda_i^k \alpha_i^k \nabla_{c_i} E(w^k)^T d_i^k.$$

Thus, from equation C.10 we obtain

$$\alpha_i^k \nabla_{c_i} E(u^k)^T d_i^k \leq -\gamma_i^l \left(\alpha_i^k\right)^2 \|d_i^k\|^2, \tag{C.15}$$

and similarly, from equations C.12, C.13, and C.14 we get, respectively,

$$\alpha_i^k \nabla_{c_i} E(u^k)^T d_i^k \geq -\gamma_i^u \left(\alpha_i^k\right)^2 \|d_i^k\|^2 \tag{C.16}$$

$$\left(\lambda_i^k \alpha_i^k - \alpha_i^k\right) \nabla_{c_i} E(v^k)^T d_i^k > 0 \tag{C.17}$$

$$\lambda_i^k \alpha_i^k \nabla_{c_i} E(w^k)^T d_i^k > -\gamma_i^l \left(\lambda_i^k \alpha_i^k\right)^2 \|d_i^k\|^2. \tag{C.18}$$

Now, from assertion i, it follows that u^k, v^k, w^k converge to the same point $(\bar{w}, \bar{c}_1, \ldots, \bar{c}_i, \ldots, \bar{c}_M)$ for $k \in K$ and $k \to \infty$. As $\{(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)\}_K$ converges and ∇E is continuous, we have that $d_i^k = -\nabla_i E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)$ is bounded, so that assertion i implies that

$$\lim_{k \to \infty, k \in K} \alpha_i^k \|d_i^k\|^2 = 0, \tag{C.19}$$

from which, recalling that λ_i^k is bounded, we obtain

$$\lim_{k \to \infty, k \in K} \lambda_i^k \alpha_i^k \|d_i^k\|^2 = 0. \tag{C.20}$$

Note that equation C.15 holds for all k and that there exists an infinite subset $K_1 \subseteq K$ such that at least one of the conditions C.16 through C.18 is satisfied for all $k \in K_1$. In all cases, either from equations C.15 and C.16, or from equations C.15 and C.17 (where $\lambda_i^k > 1$), or from equations C.15 and C.18, taking limits on the subset K_1, recalling assertion i, equations C.19 and C.20 and the continuity assumption on $\nabla_{c_i} E$, we get

$$\begin{aligned}
&\lim_{k \to \infty, k \in K_1} \nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)^T d_i^k = \\
&\lim_{k \to \infty, k \in K_1} -\nabla_{c_i} E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k)^T \nabla_{c_i} \\
&\qquad \times E(w^{k+1}, c_1^{k+1}, \ldots, c_i^k, \ldots, c_M^k) = \\
&\nabla_{c_i} E(\bar{w}, \bar{c}_1, \ldots, \bar{c}_i, \ldots, \bar{c}_M)^T \nabla_{c_i} E(\bar{w}, \bar{c}_1, \ldots, \bar{c}_i, \ldots, \bar{c}_M) = 0,
\end{aligned}$$

which contradicts equation C.9.

Acknowledgments

We are indebted to two anonymous reviewers for their useful and constructive suggestions.

References

Bertsekas, D. P. (1999). *Nonlinear programming*. Boston: Athena Scientific.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Buzzi, C., Grippo, L., & Sciandrone, M. (1999). *Convergent decomposition techniques for training RBF neural networks* (Tech. Rep. No. IASI 504). Istituto di Analisi dei Sistemi ed Informatica.

Gill, P. E., & Murray, W. (1979). *Conjugate-gradient methods for large-scale nonlinear optimization* (Tech. Rep. No. SOL 79-15). Department of Operations Research, Stanford University.

Gill, P. E., Murray, W., & Wright, M. H. (1981). *Practical Optimization*. London: Academic Press.

Grippo, L., & Sciandrone, M. (1999). Globally convergent block-coordinate techniques for unconstrained optimization. *Optimization Methods and Software, 10*, 587–637.

Grippo, L., & Sciandrone, M. (2000). On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters, 26*, 127–136.

Haykin, S. (1999). *Neural networks*. Englewood Cliffs, NJ: Prentice Hall.

Moody, J., & Darken, L. W. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation, 1*, 281–294.

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE, 78*, 1481–1497.

Powell, M. J. D. (1973). On search directions for minimization algorithms. *Mathematical Programming, 4*, 193–201.

Preschelt, L. (1994). *Proben 1—a set of neural network benchmark problems and benchmarking rules* (Tech. Rep. No. 21/94). Karlsruhe, Germany: Fakultät für Informatic, Universität Karlsruhe.

Wettschereck, D., & Dietterich, T. (1992). Improving the performance of radial basis function networks by learning locations. In J. E. Moody, S. J. Hanson, & R. P. Lippman (Eds.), *Advances in neural information processing systems, 4* (pp. 1133–1140).

Received February 3, 2000; accepted October 2, 2000.

This article has been cited by: